facebook
Artificial Intelligence Research

How to solve an MDP approximately

Matteo Pirotta

Facebook AI Research

Outline

1. Double Q-learning
2. Actor-Critic
3. Policy Performance Bound
   - NPG
   - TRPO
   - PPO
4. Regularization
   - Entropy Regularization

Q-learning Issue

Definition of Q-function

$$Q^\star(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} Q^\star(s',a')$$

$$= r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s', a_0 = a', \pi^*\right]$$

Q-learning

$$\widehat{Q}(s_t,a_t) = \widehat{Q}(s_t,a_t) + \alpha(r_t + \max_{a'} \widehat{Q}(s_{t+1},a') - \widehat{Q}(s_t,a_t))$$

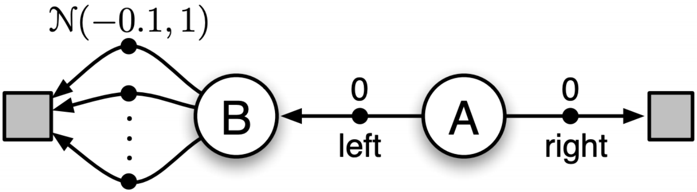
Q-learning Issue

- $\widehat{Q}$ is an incremental estimate of $Q^\star$
- We use a maximum over estimated values as an estimate of the maximum value

issue: $$\mathbb{E}[\max_{a'} \widehat{Q}(s,a')] \neq \max_{a'} \mathbb{E}[\widehat{Q}(s,a')]$$

- *This leads to a significant positive bias*

Q-learning Issue



* see Example 6.7 in [Sutton and Barto, 2018]

Double Q-learning

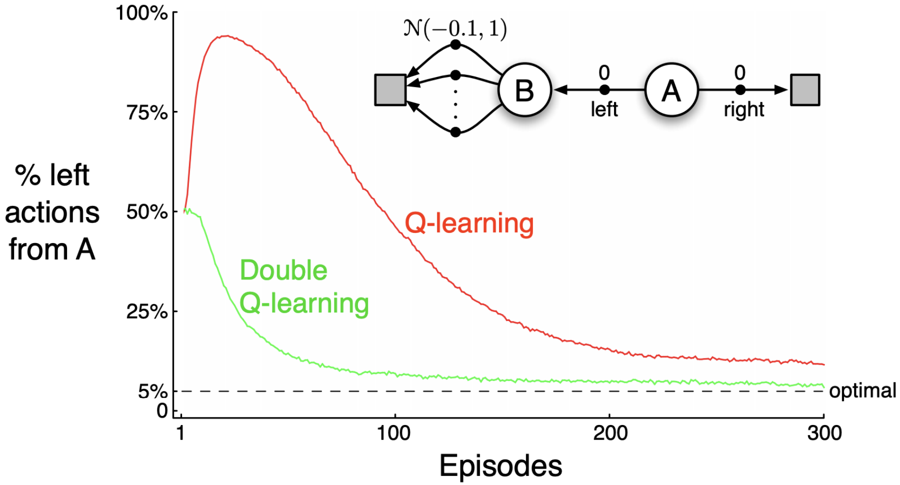
- Mitigate the over estimation problem of Q-learning
- Train simultaneously 2 Q-functions: Q_A and Q_B
- Use the maximum of Q_A to update Q_B (and vice versa)

$$Q_B(s,a) = Q_B(s,a) + \alpha \left(r + \gamma Q_B(s', \arg\max_{a'} Q_A(s',a')) - Q_B(s,a) \right)$$

Double Q-learning

Input: α, ϵ
Initialize Q_A and Q_B
for $t = 1, \ldots, T$ **do**
  Select a_t as the ϵ-greedy policy on Q_A or Q_B (or $Q_A + Q_B$)
  Observe reward r_t and next state s_{t+1}
  if *with probability* 0.5 **then**
    $a_A^+ = \arg\max_{a'} Q_A(s_{t+1}, a')$
    $Q_B(s_t, a_t) = Q_B(s_t, a_t) + \alpha \left(r + \gamma Q_B(s_{t+1}, a_A^+) - Q_B(s_t, a_t) \right)$
  else
    $a_B^+ = \arg\max_{a'} Q_B(s_{t+1}, a')$
    $Q_A(s_t, a_t) = Q_A(s_t, a_t) + \alpha \left(r + \gamma Q_A(s_{t+1}, a_B^+) - Q_A(s_t, a_t) \right)$
  end
end

Example: Double Q-Learning



* see Example 6.7 in [Sutton and Barto, 2018]

Actor-Critic

REINFORCE

- Monte-Carlo policy gradient is unbiased but *still* has high variance

REINFORCE

- Monte-Carlo policy gradient is unbiased but *still* has high variance
- Define an alternative estimate of $q^{\pi}(s, a) \implies$ actor-critic

  Critic: estimate the value function

  Actor: update the policy in the direction suggested by the critic

- It is basically *policy iteration with function approximation*

Actor-Critic

- Actor-critic algorithms maintain two sets of parameters: $\theta \mapsto \pi$, $\omega \mapsto q^{\pi}$
- *Critic can use TD(0)*

for $t = 1, \ldots, T$ **do**

$a_t \sim \pi^{\theta}(s_t, \cdot)$ and observer r_t and s_{t+1}

Compute temporal difference

$$\delta_t = r_t + \gamma q_{\omega}(s_{t+1}, a_{t+1}) - q_{\omega}(s_t, a_t)$$

Update q estimate

$$\omega = \omega + \beta \delta_t \nabla_{\omega} q_{\omega}(x_t, a_t)$$

Update policy

$$\theta = \theta + \alpha \nabla_{\theta} \log \pi_{\theta}(s_t, a_t) q_{\omega}(s_t, a_t)$$

end

TD(0) is a semi-gradient approach [Baird, 1995, Sutton, 2015]

Actor-Critic

Issues:

- $q_\omega(s,a)$ is a biased estimate of $q^{\pi_\theta}(s,a)$
- The update of θ may not follow the gradient of $\nabla_\theta J(\pi_\theta)$

Solution:

- Choose the approximation space $q_\omega(s,a)$ carefully
  $\Longrightarrow$ ***compatible function*** *approximation between* q_ω *and* π_θ

Compatible Function Approximation

Theorem

An action value function space q_ω is compatible with a policy space π_θ if

$$q_\omega(s,a) = \omega^\intercal \nabla_\theta \log \pi_\theta(s,a)$$

If ω minimizes the squared Bellman residual

$$\omega = \arg\min_\omega \mathbb{E}_{s\sim d^{\pi_\theta}} \left[\sum_a \pi_\theta(s,a)(q^{\pi_\theta}(s,a) - q_\omega(s,a))^2 \right]$$

Then

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim d^{\pi_\theta}} \mathbb{E}_{a\sim\pi_\theta} \left[\nabla_\theta \log \pi_\theta(s,a) q_\omega(s,a)\right]$$

Actor-Critic with a baseline

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s \sim d^{\pi_\theta}} \left[\sum_a \nabla_\theta \pi_\theta(s, a)(q^{\pi_\theta}(s, a) - b(s)) \right]$$

- $b(s)$ minimizes the variance
- $v^\pi(s)$ is a good choice as baseline
  - it *minimizes the variance* in average reward [Bhatnagar et al., 2009]
- $A^\pi(s, a) = q^\pi(s, a) - v^\pi(s)$ is the advantage function

Actor-Critic with advantage function (A2C)

- It is possible to estimate v^π and q^π *independently* (e.g., by TD(0))

Actor-Critic with advantage function (A2C)

- It is possible to estimate v^π and q^π *independently* (e.g., by TD(0))
- $A^\pi = q_\omega - v_\nu$ is a **biased** and **unstable** estimate

Actor-Critic with advantage function (A2C)

- It is possible to estimate v^π and q^π *independently* (e.g., by TD(0))
- $A^\pi = q_\omega - v_\nu$ is a **biased** and **unstable** estimate

Solution:

- Consider the temporal difference error

$$\delta^{\pi_\theta} = r(s,a) + \gamma v^{\pi_\theta}(s') - v^{\pi_\theta}(s)$$

Actor-Critic with advantage function (A2C)

- It is possible to estimate v^{π} and q^{π} *independently* (e.g., by TD(0))
- $A^{\pi} = q_{\omega} - v_{\nu}$ is a **biased** and **unstable** estimate

Solution:

- Consider the temporal difference error

$$\delta^{\pi_\theta} = r(s, a) + \gamma v^{\pi_\theta}(s') - v^{\pi_\theta}(s)$$

- δ^{π_θ} is an *unbiased estimate of the advantage*

$$\mathbb{E}[\delta^{\pi_\theta} | s, a] = \mathbb{E}[r(s, a) + \gamma v^{\pi_\theta}(s') | s, a] - v^{\pi_\theta}(s) = q^{\pi_\theta}(s, a) - v^{\pi_\theta}(s)$$

Actor-Critic with advantage function (A2C)

- Estimate **only** $v_\nu \mapsto \delta_\nu = r + \gamma v_\nu(s') - v_\nu(s)$

Convergence results with compatible function approximation [Bhatnagar et al., 2009]

for $t = 1, \ldots, T$ **do**

$a_t \sim \pi^\theta(s_t, \cdot)$ and observer r_t and s_{t+1}

Compute temporal difference

$$\delta_t = r_t + \gamma v_\nu(s_{t+1}) - v_\nu(s_t)$$

Update v estimate

$$\nu = \nu + \beta \delta_t \nabla_\nu v_\nu(s_t)$$

Update policy

$$\theta = \theta + \alpha \delta_t \nabla_\theta \log \pi_\theta(s_t, a_t)$$

end

From online to batch actor-critic

- So far we have observed fully online actor-critic approaches
  - The policy is updated at each step
- In some case it can be *inefficient* (e.g., for training approximators)

$\Longrightarrow$ *batching* as in supervised learning

Batch Policy Evaluation

1. Sample m trajectories $\tau_i = \{s_1, a_1, r_1, \ldots, s_{T_i}\}$ using π_θ

$$\hat{y}(s_{i,t}) = \sum_{k=t}^{t+p} \gamma^{k-t} r_{i,k} + \underbrace{\gamma^{p+1} v_\nu(s_{i,t+p+1}) \cdot \mathbb{1}\left(s_{i,t+p+1} \text{ is not terminal}\right)}_{\text{Bootstrap if not terminal}}$$

2. Use *supervised regression* on $D = \{(s_{i,t}, \hat{y}(s_{i,t}))\}$, for all i, t

$$\nu' = \arg\min_{\nu} \frac{1}{2} \sum_{(s,\hat{y}) \in D} \left(v_\nu(s) - \hat{y}\right)^2$$

☞ p is a parameter of the algorithm.
Often p is set large (to cover the entire trajectory) leading to

$$\hat{v}(s_{i,t}) = \sum_{k=t}^{T_i - 1} \gamma^{k-t} r_{i,k} + \gamma^{T_i - t} v_\nu(s_{i,T_i}) \cdot \mathbb{1}\left(s_{T_i} \text{ is not terminal}\right)$$

Batch Policy Update

1. Sample m trajectories $\tau_i = \{s_1, a_1, r_1, \ldots, s_{T_i}\}$ using π_θ
   [use the same samples for evaluation]
2. Compute an estimate of the gradient

$$\widehat{g} = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T_i} \nabla_\theta \log \pi_\theta(s_{i,t}, a_{i,t}) \delta_{i,t}$$

   where $\delta_{i,t} = r_{i,t} + \gamma v_\nu(s_{i,t+1}) - v_\nu(s_{i,t})$
3. Update the policy using gradient $\theta' = \theta + \alpha \widehat{g}$

☞ The temporal difference error is often updated using the target for policy evaluation

$$\delta_{i,t} = \widehat{y}(s_{i,t}) - v_\nu(s_{i,t})$$

Entropy Regularization

$$\max_{\pi} \left\{ J(\pi) = \mathbb{E}\left[\sum_{t=1}^{+\infty} \gamma^{t-1} r_t - \alpha \Omega(\pi(s_t, \cdot)) \right] \right\}$$

with

$$\Omega(\pi(s, \cdot)) = \sum_{a} \pi(s, a) \log \pi(s, a)$$ *negative entropy*

Entropy regularization is used to enforce randomization at the level of actions (i.e., exploration)

Randomization at the level of actions is not the best form of exploration!!!
but it is easy to implement

exercise, compute the gradient

Batched A2C

---

for $k = 1, 2, \ldots$ **do**

Generate m trajectories (τ_i) using policy π_{θ_k}

Update v

$$\nu_k = \arg\min_{\nu} \frac{1}{2} \sum_{(s,\hat{y}) \in D} (v_\nu(s) - \hat{y})^2$$

with $\mathcal{D} = (s_{i,t}, \hat{y}(s_{i,t}))_{i,t}$

Update policy

$$\delta_{i,t} = \widehat{y}(s_{i,t}) - v_{\nu_k}(s_{i,t}),$$

$$\widehat{g} = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T_i} \nabla_\theta \log \pi_\theta(s_{i,t}, a_{i,t}) \delta_{i,t} - \nabla_\theta \left(\frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T_i} \Omega(\pi_\theta(s_t, \cdot)) \right)$$

$$\theta_{k+1} = \theta_{k+1} + \alpha \widehat{g}$$

end

Sample Efficiency in Actor-Critic

Issues:

- Sample efficiency is pretty poor
- All samples need to be generated by the current policy (*on-policy learning*)
- Samples are *discarded* after a single update

Sample Efficiency in Actor-Critic

Issues:

- Sample efficiency is pretty poor
- All samples need to be generated by the current policy (*on-policy learning*)
- Samples are *discarded* after a single update

Solutions

- Use samples from other policies via *importance sampling* (*not very stable*)
- *Conservative approaches*
- Variance reduction techniques
- Newton or Quasi-newton methods

Conservative Approaches: a form of regularization

Relative Performance

Issues:

- We would like to exploit past samples
- We do not know how much to trust them
- Depends on the distribution over trajectories induced by different policies

Relative Performance

Issues:

- We would like to exploit past samples
- We do not know how much to trust them
- Depends on the distribution over trajectories induced by different policies

Performance-Difference Lemma

[Burnetas and Katehakis, 1997, Prop. 1], [Kakade and Langford, 2002, Lem. 6.1], [Cao, 2007]

For any policies $\pi, \pi' \in \Pi^{\mathsf{SR}}$

$$\begin{aligned} J(\pi') - J(\pi) &= \sum_{s,a} d^{\pi'}(s,a) A^{\pi}(s,a) \\ &= \sum_{s} d^{\pi'}(s) \sum_{a} \pi'(s,a) A^{\pi}(s,a) \end{aligned}$$

Proof

$$
\begin{aligned}
\mathbb{E}_{(s,a)\sim d^{\pi'}}[A^{\pi}(s,a)] &= \mathbb{E}_{(s,a)\sim d^{\pi'}}[q^{\pi}(s,a) - v^{\pi}(s)] \\
&= \mathbb{E}_{(s,a)\sim d^{\pi'}}[r(s,a)] + \mathbb{E}_{(s,a)\sim d^{\pi'}}\left[\gamma \sum_{y} p(y|s,a) v^{\pi}(y) - v^{\pi}(s)\right] \\
&= J(\pi') + \ \mathbb{E}_{(s,a)\sim d^{\pi'}}\left[\gamma \sum_{y} p(y|s,a) v^{\pi}(y)\right] \ - \mathbb{E}_{s\sim d^{\pi'}}[v^{\pi}(s)]
\end{aligned}
$$

Proof

$$\begin{aligned}
\mathbb{E}_{(s,a)\sim d^{\pi'}}[A^\pi(s,a)] &= \mathbb{E}_{(s,a)\sim d^{\pi'}}[q^\pi(s,a) - v^\pi(s)] \\
&= \mathbb{E}_{(s,a)\sim d^{\pi'}}[r(s,a)] + \mathbb{E}_{(s,a)\sim d^{\pi'}}\left[\gamma \sum_y p(y|s,a)v^\pi(y) - v^\pi(s)\right] \\
&= J(\pi') + \mathbb{E}_{(s,a)\sim d^{\pi'}}\left[\gamma \sum_y p(y|s,a)v^\pi(y)\right] - \mathbb{E}_{s\sim d^{\pi'}}[v^\pi(s)]
\end{aligned}$$

$$\boxed{\begin{aligned}
&= \sum_s \left(\sum_{k=0}^{+\infty} \gamma^k \mathbb{P}(s_1 \to s, k, \pi', \rho)\right) \gamma \sum_{a,y} \pi'(s,a) p(y|s,a) v^\pi(y) \\
&= \sum_y \left(d^{\pi'}(y) - \underbrace{\mathbb{P}(s_1 \to y, 0, \pi, \rho)}_{:=\rho(y)}\right) v^\pi(y)
\end{aligned}}$$

Proof

$$
\begin{aligned}
\mathbb{E}_{(s,a)\sim d^{\pi'}}[A^\pi(s,a)] &= \mathbb{E}_{(s,a)\sim d^{\pi'}}[q^\pi(s,a) - v^\pi(s)] \\
&= \mathbb{E}_{(s,a)\sim d^{\pi'}}[r(s,a)] + \mathbb{E}_{(s,a)\sim d^{\pi'}}\left[\gamma \sum_y p(y|s,a) v^\pi(y) - v^\pi(s)\right] \\
&= J(\pi') + \ \mathbb{E}_{(s,a)\sim d^{\pi'}}\left[\gamma \sum_y p(y|s,a) v^\pi(y)\right] \ - \mathbb{E}_{s\sim d^{\pi'}}[v^\pi(s)]
\end{aligned}
$$

$$
\boxed{
\begin{aligned}
&= \sum_s \left(\sum_{k=0}^{+\infty} \gamma^k \mathbb{P}(s_1 \to s, k, \pi', \rho) \right) \gamma \sum_{a,y} \pi'(s,a) p(y|s,a) v^\pi(y) \\
&= \sum_y \left(d^{\pi'}(y) - \underbrace{\mathbb{P}(s_1 \to y, 0, \pi, \rho)}_{:=\rho(y)} \right) v^\pi(y)
\end{aligned}
}
$$

$$
= J(\pi') + \sum_y d^{\pi'}(y) v^\pi(y) - \sum_y \rho(y) v^\pi(y) - \mathbb{E}_{s\sim d^{\pi'}}[v^\pi(s)]
$$

Optimization step

$$\max_{\pi'} J(\pi')$$

Optimization step

$$\max_{\pi'} J(\pi') = \max_{\pi'} J(\pi') - J(\pi)$$

Issue: as before, cannot be directly estimated using information from π

Optimization step

$$\max_{\pi'} J(\pi') = \max_{\pi'} J(\pi') - J(\pi)$$
$$= \max_{\pi'} \mathbb{E}_{(s,a)\sim d^{\pi'}} \left[A^{\pi}(s,a)\right]$$

Issue: as before, cannot be directly estimated using information from π

Optimization step

$$J(\pi') - J(\pi) = \mathbb{E}_{s\sim d^{\pi}}\left[\sum_a \pi'(s,a)A^{\pi}(s,a)\right] + \sum_s (d^{\pi'}(s) - d^{\pi}(s))\sum_a \pi'(s,a)A^{\pi}(s,a)$$

Optimization step

$$J(\pi') - J(\pi) = \mathbb{E}_{s\sim d^{\pi}}\left[\sum_a \pi'(s,a)A^{\pi}(s,a)\right] + \sum_s \underbrace{\left(d^{\pi'}(s) - d^{\pi}(s)\right)}_{?} \sum_a \pi'(s,a)A^{\pi}(s,a)$$

$$\geq \mathbb{E}_{s\sim d^{\pi}}\left[\sum_a \pi'(s,a)A^{\pi}(s,a) - \frac{\gamma\varepsilon}{(1-\gamma)^2} D_{TV}(\pi'\|\pi)[s]\right]$$

where $\varepsilon = \max_s \left|\mathbb{E}_{a\sim\pi'}[A^{\pi}(s,a)]\right|$ and

$$D_{TV}(\pi'\|\pi)[s] = \sum_a |\pi'(s,a) - \pi(s,a)|$$

Surrogate Loss

$$L_\pi(\pi') = J(\pi) + \sum_s d^\pi(s) \sum_a \pi'(s,a) A^\pi(s,a)$$

- $L_\pi(\pi) = J(\pi)$
- If parametric policies $\pi = \pi_\theta$, $\nabla_\theta L_{\pi_\theta}(\pi_\theta) = \nabla_\theta J(\pi_\theta)$

! in an interval close to π, L_π is a good surrogate for J

$\Longrightarrow$ *Conservative Policy Iteration* [Kakade and Langford, 2002]

Surrogate Loss

$$L_\pi(\pi') = J(\pi) + \sum_s d^\pi(s) \sum_a \pi'(s,a) A^\pi(s,a) \quad - \sum_s d^\pi(s) \frac{\gamma\varepsilon}{(1-\gamma)^2} D_{TV}(\pi' \| \pi)[s]$$

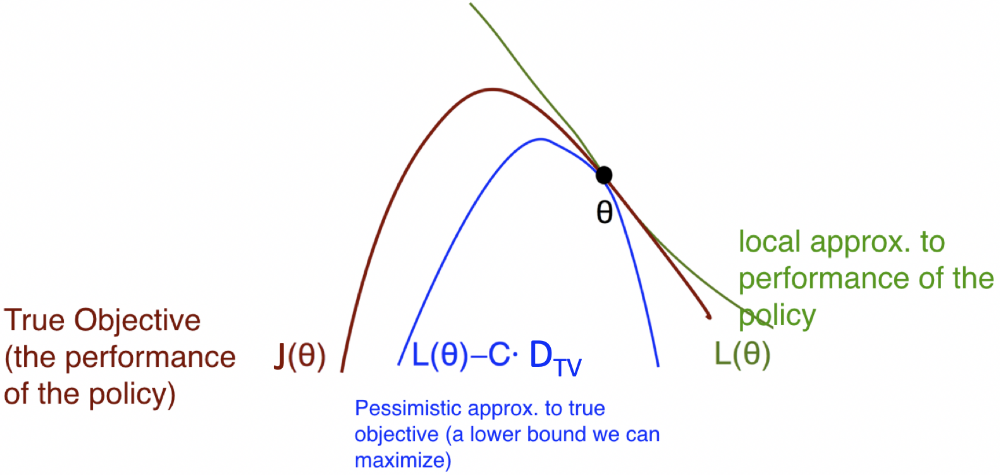
also with this

- $L_\pi(\pi) = J(\pi)$
- If parametric policies $\pi = \pi_\theta$, $\nabla_\theta L_{\pi_\theta}(\pi_\theta) = \nabla_\theta J(\pi_\theta)$

! in an interval close to π, L_π is a good surrogate for J

$\Longrightarrow$ *Conservative Policy Iteration* [Kakade and Langford, 2002]

Surrogate Loss Cont'd



Conservative Policy Iteration

- *New policy improvement schema*
  - Give current policy π_k solve

$$\max_{\pi'} \left\{ L_{\pi_k}(\pi') - C \; \mathbb{E}_{s \sim d^{\pi_k}} \left[D_{TV}(\pi' \| \pi_k)[s] \right] \right\}$$

Conservative Policy Iteration

- *New policy improvement schema*
  - Give current policy π_k solve

$$\max_{\pi'} \left\{ L_{\pi_k}(\pi') - \boldsymbol{C}\, \mathbb{E}_{s\sim d^{\pi_k}} \left[D_{TV}(\pi' \| \pi_k)[s] \right] \right\} \geq 0$$

Conservative Policy Iteration

- *New policy improvement schema*
  - Give current policy π_k solve

$$\boldsymbol{J(\pi') - J(\pi_k)} \geq \max_{\pi'} \left\{ L_{\pi_k}(\pi') - \boldsymbol{C}\, \mathbb{E}_{s \sim d^{\pi_k}} \left[D_{TV}(\pi' \| \pi_k)[s] \right] \right\} \geq \mathbf{0}$$

Conservative Policy Iteration

- *New policy improvement schema*
  - Give current policy π_k solve

$$\boldsymbol{J(\pi') - J(\pi_k)} \geq \max_{\pi'} \left\{ L_{\pi_k}(\pi') - \boldsymbol{C} \; \mathbb{E}_{s \sim d^{\pi_k}} \left[D_{TV}(\pi' \| \pi_k)[s] \right] \right\} \geq \mathbf{0}$$

$\Longrightarrow$ *Monotonic performance improvement*

Conservative Policy Iteration

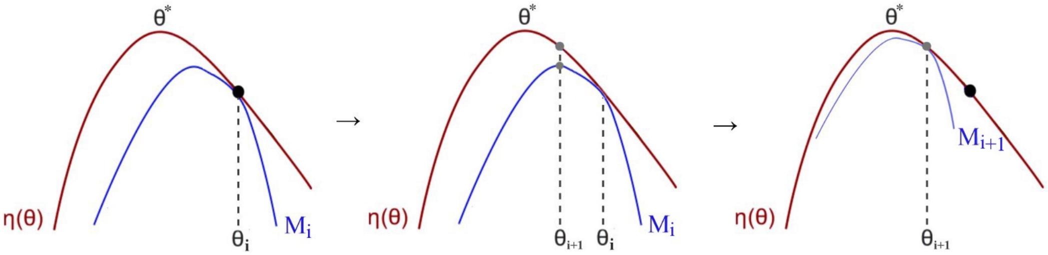
- *New policy improvement schema*
  - Give current policy π_k solve

$$\boldsymbol{J(\pi') - J(\pi_k)} \geq \max_{\pi'} \left\{ L_{\pi_k}(\pi') - \boldsymbol{C} \, \mathbb{E}_{s \sim d^{\pi_k}} \left[D_{TV}(\pi' \| \pi_k)[s] \right] \right\} \geq \mathbf{0}$$

$\Longrightarrow$ *Monotonic performance improvement*

Several approaches have been proposed [e.g., Kakade and Langford, 2002, Perkins and Precup, 2002, Gabillon et al., 2011, Wagner, 2011, 2013, Pirotta et al., 2013b, Scherrer et al., 2015, Schulman et al., 2015]

Idea



$$\eta(\theta) = \mathbb{E}\left[\sum_{t=1}^{\infty} r_t | \pi_\theta\right]$$ and M is the lower bound

Source

Approximate Monotone Improvement

- The objective can be estimated using rollouts from the most recent policy
- Updates respect a notion of distance in the policy space!

This is the basis for many algorithms!

Toward Practical Algorithm

- Optimizing the total variation $D_{TV}(\pi' \| \pi)$ may be *difficult*
- Relax the problem using *Pinsker's inequality* [Csiszar and Körner, 2011]

$$D_{TV}(\pi' \| \pi) \leq \sqrt{2 D_{KL}(\pi' \| \pi)}$$

Further Steps toward Practical Algorithms

- C provided by theory is quite high (*too conservartive*)
- Replace regularization with constraint (*trust region*) (e.g., REPS [Peters et al., 2010])

$$\pi_{k+1} = \arg\max_{\pi'} L_\pi(\pi')$$
$$\text{s.t. } \mathbb{E}_{s\sim d^\pi}[D_{KL}(\pi' \| \pi)] \le \delta$$

Further Steps toward Practical Algorithms

- C provided by theory is quite high (*too conservartive*)
- Replace regularization with constraint (*trust region*) (e.g., REPS [Peters et al., 2010])

$$\pi_{k+1} = \arg\max_{\pi'} L_\pi(\pi')$$
$$\text{s.t. } \mathbb{E}_{s\sim d^\pi}[D_{KL}(\pi'\|\pi)] \leq \delta$$

- Importance weighting

$$\mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi'}[A^\pi(s,a)] = \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim z}\left[\frac{\pi'(s,a)}{z(s,a)}A^\pi(s,a)\right]$$

Further Steps toward Practical Algorithms

- C provided by theory is quite high (*too conservartive*)
- Replace regularization with constraint (*trust region*) (e.g., REPS [Peters et al., 2010])

$$\pi_{k+1} = \arg\max_{\pi'} L_\pi(\pi')$$
$$\text{s.t. } \mathbb{E}_{s\sim d^\pi}[D_{KL}(\pi'\|\pi)] \le \delta$$

- Importance weighting

$$\mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim\pi'}[A^\pi(s,a)] = \mathbb{E}_{s\sim d^\pi}\mathbb{E}_{a\sim z}\left[\frac{\pi'(s,a)}{z(s,a)}A^\pi(s,a)\right]$$

$\Longrightarrow$ Natural Policy Gradient (NPG) [Kakade, 2002]
$\Longrightarrow$ Trust-Region Policy Optimization (TRPO) [Schulman et al., 2015]

Practical Algorithms

Gradient Descent

Steepest descent direction of a function $h(\theta) \to -\nabla h(\theta)$

- It yields the *most reduction* in h per unit of change in θ
- Change is measured using the standard *Euclidean norm* $\|\cdot\|$

$$\frac{-\nabla h}{\|\nabla h\|} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \arg\min_{d:\|d\| \leq \epsilon} \{h(\theta + d)\}$$

Gradient Descent

Steepest descent direction of a function $h(\theta) \to -\nabla h(\theta)$

- It yields the *most reduction* in h per unit of change in θ
- Change is measured using the standard *Euclidean norm* $\|\cdot\|$

$$\frac{-\nabla h}{\|\nabla h\|} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \arg\min_{d : \|d\| \leq \epsilon} \{h(\theta + d)\}$$

Is the Euclidean norm the best metric?
Can we use an alternative definition of (*local*) distance?

(Example: gradient descent is not affine invariant)

Natural Gradient

- In Riemannian space, the distance is defined as

$$d^2(v, v + \delta v) = \delta v^\intercal G(v) \delta v^\intercal$$

where G is the *metric tensor*

Natural Gradient [Amari, 1998]

The steepest descent in a Riemannian is given by

$$\widetilde{\nabla} h(\theta) = G(\theta)^{-1} \nabla h(\theta)$$

KL divergences and the Fisher information matrix

- The Kullback Leibler divergence can be *approximated* by the Fisher information matrix (2nd order Taylor approximation)

$$D_{KL}(p(x|\theta)\|p(x|\theta+\Delta\theta)) = \Delta\theta^{\mathsf{T}} F(\theta)\Delta\theta + O(\Delta\theta^3)$$

  where $F(\theta)$ is the *Fisher Information Matrix (FIM)*

$$F(\theta) = \mathop{\mathbb{E}}_{x\sim p(\cdot|\theta)}\left[\nabla \log p(x|\theta)\, \nabla \log p(x|\theta)^{\mathsf{T}}\right]$$

- Captures information how a parameter influences the distribution

Natural Gradient in Distribution Space

- NG uses the *Fisher information matrix as metric*
  - Find direction maximally correlated with gradient
  - Constraint: (approximated) KL should be bounded

$$\widetilde{\nabla} h(\theta) = \arg\max_{\Delta\Theta} \Delta\theta^{\mathsf{T}} \nabla h(\theta)$$

$$\text{st.} D_{KL}(p(x|\theta) \| p(x|\theta + \Delta\theta) \approx \Delta\theta^{\mathsf{T}} F(\theta) \Delta\Theta \leq \epsilon$$

- $\widetilde{\nabla} h(\theta) = F(\theta)^{-1} \nabla h(\theta)$
- $\widetilde{\nabla} h$ is be *invariant* to the choice of parameterization

Natural Policy Gradient

$$\pi_{k+1} = \arg\max_{\pi'} \underbrace{\mathbb{E}_{s\sim d^{\pi}} \mathbb{E}_{a\sim z} \left[\frac{\pi'(s,a)}{z(s,a)} q^{\pi}(s,a) \right]}_{:=\mathcal{L}_{\pi_k}(\pi')}$$

$$\text{s.t. } \underbrace{\mathbb{E}_{s\sim d^{\pi}} [D_{KL}(\pi' \| \pi)]}_{:=\overline{D}_{KL}(\pi' \| \pi)} \leq \delta$$

How to solve it? Do it approximately

$$\mathcal{L}_{\theta_k}(\theta) \approx L_{\theta_k}(\theta_k) + g^{\mathsf{T}}(\theta - \theta_k)$$

$$\overline{D}_{KL}(\theta \| \theta_k) \approx \frac{1}{2}(\theta - \theta_k)^{\mathsf{T}} F(\theta)(\theta - \theta_k)$$

where $g = \nabla_{\theta} \mathcal{L}_{\theta_k}(\theta)$ and $F(\theta) := \nabla^2_{\theta} \overline{D}_{KL}(\theta \| \theta_k)$ is the FIM.

Natural Policy Gradient

The approximate problem is thus

$$\theta_{k+1} = \arg\max_{\theta} g^{\mathsf{T}}(\theta - \theta_k)$$

$$\text{s.t. } \frac{1}{2}(\theta - \theta_k)^{\mathsf{T}} F (\theta - \theta_k) \leq \delta$$

whose solution is given by:

$$\theta_{k+1} = \theta_k + \underbrace{\sqrt{\frac{2\delta}{g^{\mathsf{T}} F^{-1} g}}}_{\text{step size}} \underbrace{F^{-1} g}_{\text{natural gradient}}$$

Algorithms [Kakade, 2002, Peters and Schaal, 2008]

Natural Policy Gradient

Initialize policy parameter θ_0
for $k = 1, 2, \ldots$ **do**

Collect trajectories $\mathcal{D}_k$ using policy $\pi_k = \pi(\theta_k)$
Estimate advantage function using any algorithm
Compute

- policy gradient $\widehat{g}_k$ (using advantage estimate)
- KL-divergence Hessian / Fisher information matrix $\widehat{F}_k$

Compute new policy using natural gradient

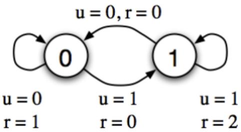
$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{\widehat{g}_k^{\mathsf{T}} \widehat{F}_k^{-1} \widehat{g}_k}} \widehat{F}_k^{-1} \widehat{g}_k$$

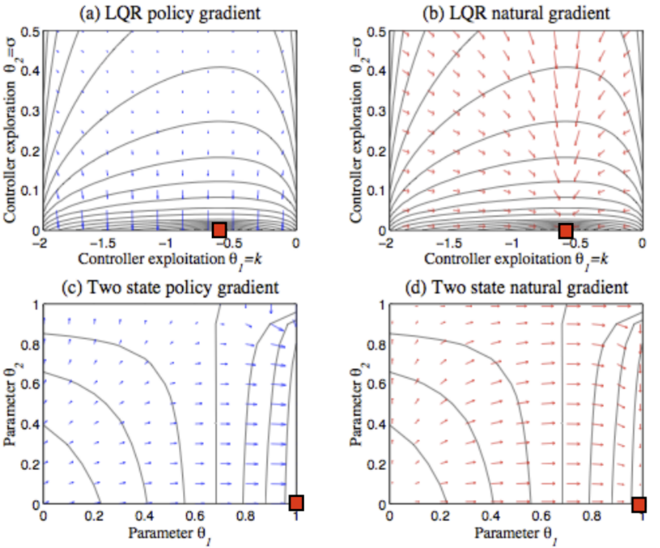
end

Linear Quadratic Regulation

$$x_{t+1} = Ax_t + Bu_t$$
$$u_t \sim \pi(u|x_t) = \mathcal{N}(u|kx_t, \sigma)$$
$$r_t = -x_t^T Q x_t - u_t^T R u_t$$

Two-State Problem





[Peters et al. 2003, 2005]

The standard gradient reduces the exploration too quickly!

source: Policy Search: Methods and Applications, Peters and Neumann video

Truncated Natural Policy Gradient

Issues:

- $\theta \in \mathbb{R}^d$, d can be very large (e.g., thousands or millions)
- H or F have dimension d^2
- matrix inversion is $\mathcal{O}(d^3)$

Truncated Natural Policy Gradient

Issues:

- $\theta \in \mathbb{R}^d$, d can be very large (e.g., thousands or millions)
- H or F have dimension d^2
- matrix inversion is $\mathcal{O}(d^3)$

Solution:

- Use conjugate gradient to compute $F^{-1}g$ without inverting F [Pascanu and Bengio, 2013]
- With j iterations, CG solves systems of equations $Hx = g$ for x by finding projection onto Krylov subspace (i.e., $span(g, Hg, \ldots H^{j-1}g)$)

$\Longrightarrow$ *Truncated Natural Policy Gradient*

Truncated Natural Policy Gradient

Issues:

- $\theta \in \mathbb{R}^d$, d can be very large (e.g., thousands or millions)
- H or F have dimension d^2
- matrix inversion is $\mathcal{O}(d^3)$

Solution:

- Use conjugate gradient to compute $F^{-1}g$ without inverting F [Pascanu and Bengio, 2013]
- With j iterations, CG solves systems of equations $Hx = g$ for x by finding projection onto Krylov subspace (i.e., $span(g, Hg, \ldots H^{j-1}g)$)

$\Longrightarrow$ *Truncated Natural Policy Gradient*

Other solutions are possible: see ACKTR [Wu et al., 2017], [Ollivier, 2017]

Trust Region Policy Optimization

Issues with NPG:

- Might not be robust to trust region size δ
- Due to approximation, KL-constraint might be violated

Solution:

- Force improvement in surrogate loss ($\mathcal{L}_{\theta_k}(\theta_{k+1}) \geq 0$)
- Enforce KL-constraint

Trust Region Policy Optimization

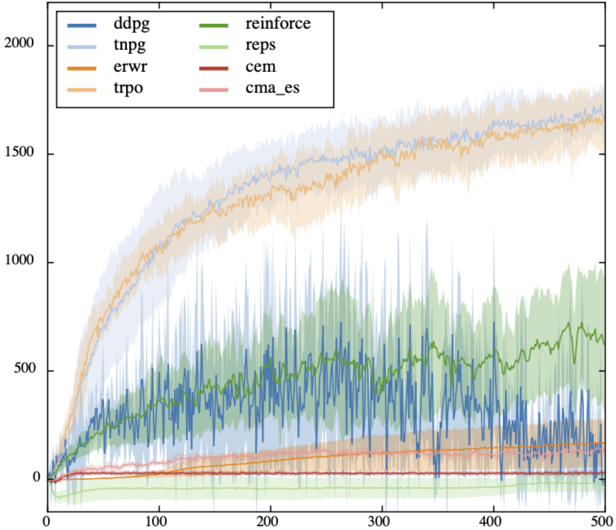
How?
Backtracking line search with exponential decay (decay coeff $\alpha \in (0, 1)$, budget L)

Compute NPG step $\Delta_k =$
For $j = 0, \dots L$ Compute update $\theta = \theta_k + \alpha^j \Delta_k$
if $\mathcal{L}_{\theta_k}(\theta) > 0$ *and* $\overline{D}_{KL}(\theta \| \theta_k) \leq \delta$ **then**
  accept update and $\theta_{k+1} = \theta_k + \alpha^j \Delta_k$
  break
end

In practice, TRPO is implemented as (T)NPG plus line search.

Example: Walker-2d

[Duan et al., 2016]



video1
video3
video2

Proximal Policy Optimization

[Schulman et al., 2017]

- Avoid to compute the natural gradient
- Approximate the KL constraint

Proximal Policy Optimization

[Schulman et al., 2017]

- Avoid to compute the natural gradient
- Approximate the KL constraint

1. *Adaptive KL Penalty*
   - Consider regularized optimization problem

$$\theta_{k+1} = \arg\max_{\theta} L_{\theta_k}(\theta) - \lambda_k \mathbb{E}[D_{KL}(\theta \| \theta_k)]$$

   - Adapt λ_k to enforce KL constraint

$$\lambda_{k+1} = \begin{cases} 2\lambda_k & \text{if } \overline{D}_{KL}(\theta \| \theta_k) \geq 1.5\delta \\ \lambda_k/2 & \text{if } \overline{D}_{KL}(\theta \| \theta_k) \leq \delta/1.5 \\ \lambda_k & \text{otherwise} \end{cases}$$

Proximal Policy Optimization

[Schulman et al., 2017]

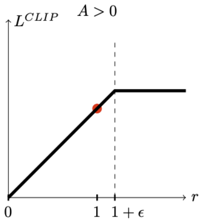
2 *Clipped Objective*

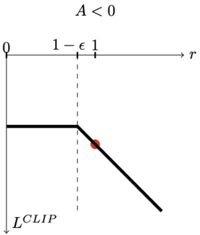
- Recall surrogate objective

$$L_{\pi}^{\text{IS}}(\pi') = \mathbb{E}_{s\sim d^{\pi}}\mathbb{E}_{a\sim\pi}\left[\frac{\pi'(s,a)}{\pi(s,a)}A^{\pi}(s,a)\right] = \mathbb{E}_{s\sim d^{\pi}}\mathbb{E}_{a\sim\pi}\left[r_{sa}(\pi')A^{\pi}(s,a)\right]$$

- Form a lower bound via clipped importance ratios

$$L_{\pi}^{\text{CLIP}}(\pi') = \mathbb{E}_{s\sim d^{\pi}}\mathbb{E}_{a\sim\pi}\left[\min\left\{r_{sa}(\pi')A^{\pi}(s,a), \text{clip}(r_{sa}(\pi'), 1-\epsilon, 1+\epsilon)A^{\pi}(s,a)\right\}\right]$$





- $\pi_{k+1} = \arg\max_{\pi} L_{\pi_k}^{\text{CLIP}}(\pi)$

PPO with Adaptive KL Penalty

Input: policy θ_0, KL penalty β_0, KL-divergence δ
for $k = 1, \ldots$ **do**
  Collect trajectories $\mathcal{D}_k$ using policy $\pi_k = \pi(\theta_k)$
  Estimate advantage or q-function using any algorithm
  Compute

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \overline{D}_{KL}(\theta \| \theta_k)$$

  by gradient descent
  if $\overline{D}_{KL}(\theta_{k+1} \| \theta_k) \geq 1.5\delta$ **then**
    $\beta_{k+1} = 2\beta_k$
  end
  if $\overline{D}_{KL}(\theta_{k+1} \| \theta_k) \geq \delta / 1.5$ **then**
    $\beta_{k+1} = \beta_k / 2$
  end
end

Initial β_0 not important. Some iteration may violate KL constraint, mostly not!

PPO with Clipping

Input: policy θ_0, clipping ϵ

for $k = 1, \dots$ **do**

Collect trajectories $\mathcal{D}_k$ using policy $\pi_k = \pi(\theta_k)$

Estimate advantage or q-function using any algorithm

Compute

$$\theta_{k+1} = \arg\max_{\theta} L^{\mathsf{CLIP}}_{\theta_k}(\theta)$$

where

$$L^{\mathsf{CLIP}}_{\pi}(\pi') = \mathbb{E}_{\tau\sim\pi_k}\left[\sum_{t=1}^{T} \min\left\{r_t(\theta)\widehat{A}_t^{\pi_k}, \mathsf{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_t^{\pi_k}\right\}\right]$$

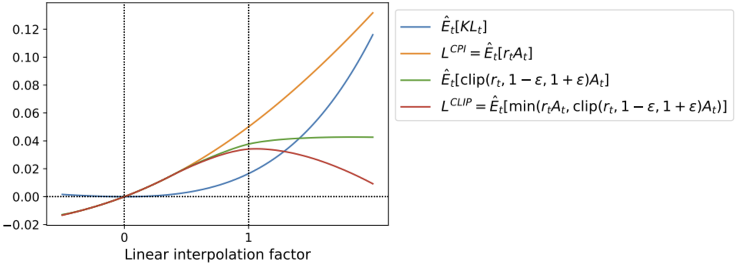
end

Proximal Policy Optimization

[Schulman et al., 2017]

- Clipping prevents policy from moving too much away from θ_k
- Seems to work as well as PPO with KL penalty
- Much simpler to implement

How does it work?



Various objectives as a function of function of α between θ_k and θ_{k+1}

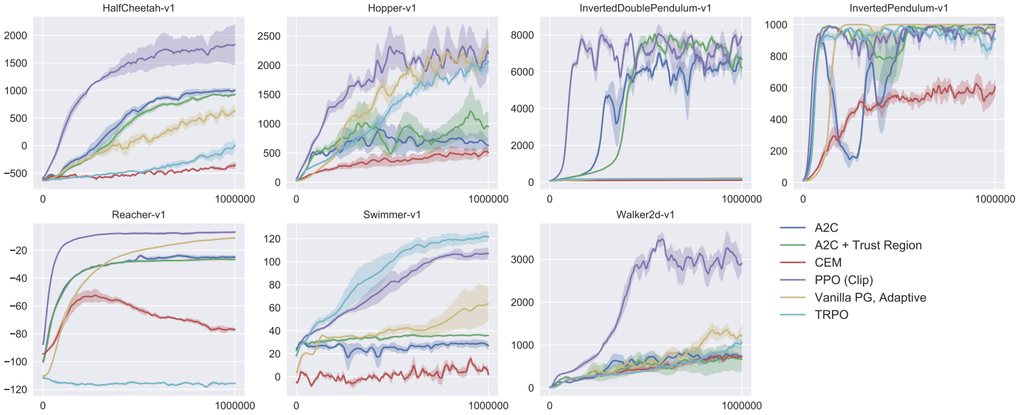


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.

video
Rubik's cube

Softmax Operator

$$v^\star(s) = \max_a \left\{ r(s,a) + \gamma \sum_y p(y|s,a) v^\star(y) \right\}$$

replace $\max$ with "*softmax*" operator

$$v^\star(s) = \frac{1}{\eta} \log \left(\sum_a \exp \left[\eta \left(r(s,a) + \gamma \sum_y p(y|s,a) v^\star(y) \right) \right] \right)$$

[Marcus et al., 1997, Ruszczyński, 2010, Ziebart et al., 2010, Ziebart, 2010, Braun et al., 2011, Azar et al., 2012, Rawlik et al., 2012, Fox et al., 2016, Asadi and Littman, 2017, Haarnoja et al., 2017, Schulman et al., 2017, Nachum et al., 2017]

Entropy Regularization

$$\max_{\pi} \left\{ J(\pi) = \mathbb{E}\left[\sum_{t=1}^{+\infty} \gamma^{t-1} r_t - \alpha \Omega(\pi(s_t, \cdot)) \right] \right\}$$

The two approaches are connected by Lagrangian duality when

$$\Omega(\pi(s, \cdot)) = \sum_a \pi(s, a) \log \pi(s, a)$$ *negative entropy*

Entropy Regularization

$$\max_{\pi} \left\{ J(\pi) = \mathbb{E}\left[\sum_{t=1}^{+\infty} \gamma^{t-1} r_t - \alpha \Omega(\pi(s_t, \cdot)) \right] \right\}$$

The two approaches are connected by Lagrangian duality when

$$\Omega(\pi(s, \cdot)) = \sum_a \pi(s, a) \log \pi(s, a) \qquad \textit{negative entropy}$$

Results: [Neu et al., 2017]

- Existence and uniqueness
- Well-defined contractive DP operator
- Policy Gradient Theorem

Entropy Regularization

Optimal policy:

$$\pi^\star(s,a) \propto \exp\left[\eta\left(r(s,a) + \gamma \mathbb{E}_s'[v^\star(s')]\right)\right]$$

Note:

$$q^\pi(s,a) = r(s,a) + \gamma \sum_y p(y|s,a) v^\pi(y)$$

$$v^\pi(s) = \mathbb{E}_{a\sim\pi}[q^\pi(s,a)] - \Omega(\pi(s,\cdot))$$

Soft-Actor Critic

1. Train the value function v

$$\arg\min_{\psi} \in \mathbb{E}_{s_t \sim H}\left[\frac{1}{2}\left(v_\psi(s_t) + \mathbb{E}_{a_t \sim \pi_\phi}[q_\theta(s_t, a_t) - \log \pi_\phi(s_t, a_t)]\right)^2\right]$$

2. Train the action-value function q^π

$$\arg\min_{\theta} \mathbb{E}_{(s,a) \in H}\left[\frac{1}{2}\left(q_\theta(s_t, a_t) - (r(s_t, a_t) + \gamma \mathbb{E}[v_{\overline{\psi}}(s')])\right)^2\right]$$

! fix the target network (e.g., DQN) $\rightarrow$ increase stability / break dependences

3. Fit the new policy

$$\arg\min_{\phi} \mathbb{E}_{s \in H}\left[D_{KL}(\pi_\psi \| \exp[\eta q_\psi]/Z)[s]\right]$$

Path-Consistency Learning

[Nachum et al., 2017]

Suppose the MDP is deterministic (otherwise take a conditional expectation w.r.t. to history)

For any $v^\star, \pi^\star$ optimizing the regularized objective

$$v^\star(s) - \gamma v^\star(s') = r(s,a) - \eta \log \pi^\star(s,a)$$

$$v^\star(s_1) - \gamma^{t-1} v^\star(s_t) = \sum_{t=1}^{t-1} \gamma^{i-1} \left(r(s_i,a_i) - \eta \log \pi^\star(s_i,a_i)\right)$$

! if (π, v) satisfies the *path consistency* for every (s,a), then $\pi = \pi^\star$ and $v = v^\star$

Path-Consistency Learning

- Maintain two sets of parameters (ϕ, θ): $\theta \mapsto \pi_\theta$, $\phi \mapsto v_\phi$
- Minimize the consistency error

$$\min_{\phi,\theta} O_{PCL}(\phi, \theta, H) = \sum_{s_{i:i+d} \in E_H} \frac{1}{2} C(s_{i:i+d}, \phi, \theta)^2$$

where E_H is the set of (sub)trajectories and

$$C(s_{i:i+d}, \phi, \theta) = -v_\phi(s_i) + \gamma^d v_\phi(s_{i+d}) + \sum_{j=0}^{d-1} \gamma^j \left(r(s_{i+j}, a_{i+j}) - \eta \log \pi_\theta(s_{a+j}, a_{i+j})\right)$$

Path-Consistency Learning

- Maintain two sets of parameters (ϕ, θ): $\theta \mapsto \pi_\theta$, $\phi \mapsto v_\phi$
- Minimize the consistency error

$$\min_{\phi,\theta} O_{PCL}(\phi, \theta, H) = \sum_{s_{i:i+d} \in E_H} \frac{1}{2} C(s_{i:i+d}, \phi, \theta)^2$$

where E_H is the set of (sub)trajectories and

$$C(s_{i:i+d}, \phi, \theta) = -v_\phi(s_i) + \gamma^d v_\phi(s_{i+d}) + \sum_{j=0}^{d-1} \gamma^j \left(r(s_{i+j}, a_{i+j}) - \eta \log \pi_\theta(s_{a+j}, a_{i+j}) \right)$$

In practice:

- Use replay buffer
- Update incrementally $\implies$ semi-batch

Path-Consistency Learning

- Maintain two sets of parameters (ϕ, θ): $\theta \mapsto \pi_\theta$, $\phi \mapsto v_\phi$
- Minimize the consistency error

$$\min_{\phi,\theta} O_{PCL}(\phi, \theta, H) = \sum_{s_{i:i+d} \in E_H} \frac{1}{2} C(s_{i:i+d}, \phi, \theta)^2$$

where E_H is the set of (sub)trajectories and

$$C(s_{i:i+d}, \phi, \theta) = -v_\phi(s_i) + \gamma^d v_\phi(s_{i+d}) + \sum_{j=0}^{d-1} \gamma^j \left(r(s_{i+j}, a_{i+j}) - \eta \log \pi_\theta(s_{a+j}, a_{i+j}) \right)$$

In practice:

- Use replay buffer
- Update incrementally $\implies$ semi-batch

Can be extended to different regularizers (e.g., Shannon entropy, Tsallis entropy [Chow et al., 2018])

Summary

- Double Q-learning
- Actor-Critic with Advantage Function
- Conservative Approaches
- Natural Policy Gradient, TRPO and PPO

Thank you!

facebook
Artificial Intelligence Research

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor-critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

Apostolos N Burnetas and Michael N Katehakis. Optimal adaptive policies for markov decision processes. *Mathematics of Operations Research*, 22(1):222–255, 1997.

X.R. Cao. *Stochastic Learning and Optimization: A Sensitivity-Based Approach*. International Series on Discrete Event Dynamic Systems, v. 17. Springer US, 2007. ISBN 9780387690827.

Yinlam Chow, Ofir Nachum, and Mohammad Ghavamzadeh. Path consistency learning in tsallis entropy regularized mdps. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 978–987. PMLR, 2018.

Imre Csiszar and János Körner. *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press, 2011.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274, 2002.

Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *NIPS*, pages 2772–2782, 2017.

Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *CoRR*, abs/1705.07798, 2017.

Yann Ollivier. True asymptotic natural gradient optimization. *arXiv preprint arXiv:1712.08449*, 2017.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Richard Sutton. Introduction to reinforcement learning with function approximation. Technical report, Tutorial at the Conference on Neural Information Processing Systems, 2015.

Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT press Cambridge, 2 edition, 2018.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5279–5288, 2017.